Design and Implementation of a Virtual Information System for Agile Manufacturing *

Liugen Song, Rakesh Nagi†
Department of Industrial Engineering, 342 Bell Hall
State University of New York at Buffalo, Buffalo, NY 14260

Abstract

In the new and emerging Agile Manufacturing paradigm, where multiple firms cooperate under flexible virtual enterprise structures, there exists much need for a mechanism to manage and control information flow among collaborating partners. In response to this pressing need, this paper addresses the design and implementation of an agile manufacturing information system integrating manufacturing databases dispersed at various partner sites. We propose a framework in which: (1) Information is modeled in a hierarchical fashion using Object Oriented Methodology (OOM). (2) Information transactions are specified by the workflow hierarchy consisting of partner workflows. (3) Information flow between partners is controlled by a set of distributed Workflow Managers (WM) interacting with partner knowledge bases, which reflect partner specific information control rules on internal data exchange, as well as inter-partner mutual protocols for joint partner communications. (4) The prototype system is accomplished using the world wide web based on a Client-Server architecture. The overall approach and system provides within a dynamic environment, where virtual partnerships are synthesized in response to specific business initiatives, a dynamic and flexible mechanism to support partner information exchange and to keep the dispersed information consistent.

Keywords: Agile Manufacturing Information Systems, Manufacturing Database Integration.

* *This work has been supported by the National Science Foundation under grant DMI-962409*
† IIE Member. To whom correspondence should be addressed. Email: nagi@eng.buffalo.edu

1 Introduction

Facing the competitive market, industrial manufacturers are hard pressed to adopt novel strategies and technologies to enhance product quality, to cut manufacturing cost and to reduce product lead time. On the basis of modern information technology, agile manufacturing focuses on enhancing the competitiveness by cooperative working. Cooperation is the philosophy behind this strategy (Nagel and Dove, 1993).

Agile manufacturing is attracting more and more attention from both academic and industrial communities. Extensive programs are being conducted on relevant issues to propagate agile manufacturing concepts, to build agile enterprise prototypes and to realize an agile industry eventually. The Agile Manufacturing Enterprise Forum (AMEF) at the Iacocca Institute of Lehigh University was created to disseminate the ideas of agile manufacturing and to increase the pace and scope of the transition to an agile manufacturing industry (Nagel and Dove, 1993). The Agile Manufacturing Initiative aims to develop, demonstrate, and evaluate the advanced design, manufacturing and business transaction processes in the agile environment *(http://www.agilityforum.org)*. The Concurrent Technologies Corp. (CTC) is developing an agile manufacturing testbed to provide Department of Defense (DoD) with increased weapon system readiness and added system mobility (Pandiarajan and Patun, 1994). The Agile Aerospace Manufacturing Research Center (AAMRC) at the University of Texas at Arlington is conducting research on agile business practices, business process identification and characterization, and enabling technologies. The Manufacturing Research, Education and Outreach Program at the University of Illinois at Urbana-Champaign is developing computer integrated manufacturing and machine tool systems. The Electronic Agile Manufacturing Research Institute (EAMRI) at Rensselaer Polytechnic Institute (RPI) is focusing on electronics product realization in distributed manufacturing environments using improved information infrastructures and architectures.

Agile manufacturing makes use of modern information technology to form virtual enterprises, which agilely respond to the changing market demands. A virtual enterprise, different from a traditional enterprise, is constructed by partners from different companies, who collaborate with each other to design and manufacture high quality and customized products. It is product-oriented, team-collaboration styled, and featured as fast and flexible. Frequent and dynamic interactions among partners in agile manufacturing enterprises entail the crucial role of an Agile Manufacturing Information System (AMIS). It is up to the AMIS to provide partners with integrated and consistent information, as well as to manage partner transactions accessing the information.

Heterogeneity, autonomy and geographical distribution of manufacturing partners bring new challenges to the development of an integrated AMIS. The main issues are the following (see also Section 3): (1) Partner information interoperability across companies. Product data and knowledge are segmented and distributed across the distributed sites of partners who need to access product information frequently and dynamically. Information sharing between distributed databases, as well as inter-partner transaction management are essential to support agile product development. (2) Information consistency across partners in the virtual enterprise. Different from a traditional enterprise where data is owned by the company itself, a virtual enterprise has multiple data ownership. Mutual agreements or protocols are needed to resolve data ownership and keep multiple replications of the same data consistent between companies. (3) Partner policy independence and autonomy maintenance. Agile partners have full autonomies, in the sense that there are no hierarchical organizations in an agile enterprise. Products are developed by collaborations performed by peer coworkers from autonomous partner companies.

Information and information transaction management in a heterarchical agile enterprise is different from that of a traditional hierarchical manufacturing enterprise. AMIS needs to retain partner autonomies. (4) Partner heterogeneity accommodation. Heterogeneity in partner computer hardwares, database systems and applications is another obstacle in achieving partner engineering information sharing and concurrent transaction management. Existing application software and databases in design/manufacturing lack compatibility and portability. AMIS must accommodate this heterogeneity. (5) Open and dynamic system architecture. Agile enterprises are virtual and product oriented, they are formed dynamically and may not exist after the product's developmental life-time. They are different from traditional manufacturing enterprises, which usually have a relatively fixed organizational structure. AMIS must provide a flexible and open architecture to host partners dynamically.

We aim to develop an AMIS framework which meets the AMIS requirements. Our objectives are listed as the following. (1) Exploring the requirements on information sharing and manipulation for agile partners to develop customized products cooperatively. (2) Designing a conceptual agile manufacturing information framework – the Virtual Information System for Agile Manufacturing (VISAM), which consists of Partner UNit Information Systems (PUNIS) and manages information and information transactions in a consistent way, so as to satisfy these requirements. (3) Building the VISAM framework by developing two hierarchies: (i) the data hierarchy, and (ii) the transaction workflow hierarchy, using system development tools. (4) Developing a Knowledge Base which stores expert rules reflecting local partner policies and inter-partner protocols to support partner transaction management. (5) Implementing a prototype virtual manufacturing enterprise information system for illustrative purposes. (6) Providing a general procedure for current manufacturers to build individual agile manufacturing information systems, to form aggregated agile manufacturing information systems for virtual enterprises, and to realize collaborative product development.

In this paper, we focus on the requirements, design and implementation issues of the agile manufacturing information system. Our interest is to draw attention to this important need in agile industry, and to establish the need for information system design based on agile manufacturing business practices in a manner that least disrupts existing company operations. We strongly believe that the information system should be designed to suit business practice and not *vice-versa* if we are to fully realize the potential of an agile industry. The first task naturally in this process is to thoroughly understand the functional and performance requirements that the agile environment places on the information system. Even while this requirements list is evergrowing and is conditioned on the specific joint-enterprise or industry instance, we have, in this paper, attempted to outline some of the most important and ubiquitous ones. On this basis, we then present an AMIS framework which aims at satisfying most of the requirements. We also identify the requirements that are out of the scope of this framework and are probably left to low-level system implementation or other research areas. We argue the relevance of the building blocks or modules of the overall system with respect to main issues and requirements of an AMIS. In order to remain focused, these modules are described here in brief and the reader is referred to Song (1996) for detailed explanations. We also present examples of how this framework supports simple information sharing scenarios, and discuss the prototype system implemented in brief. Our ultimate aim is towards providing current manufacturers a general procedure to build individual agile manufacturing information systems based on the VISAM framework, and enabling them to participate in an agile industry.

The remaining paper is organized as follows. In Section 2, we review the literature relevant to various topics that the Agile Manufacturing Information System concepts draw upon. Section 3 is devoted to the important AMIS requirements. An overview of the proposed framework, VISAM is presented in Section 4. The object-oriented information viewed as an Information Hierarchy and the corresponding Workflow Hierarchy that manages this information are detailed in Section 5. The Knowledge Based System for information control and management is presented in Section 6. A brief description of the prototype system implementation is included in Section 7. Finally, we conclude with directions for future work in Section 8.

2 Literature Review

This section is devoted to the review of previous work related to: (i) manufacturing information integration, (ii) collaborative architecture engineering construction design, and (iii) system development tools. All these topics are related to the design and implementation of a virtual information system for agile manufacturing.

2.1 Manufacturing Information Integration

Manufacturing information integration has been the focus of extensive research and development, both in data integration and in knowledge integration. To realize aggregated functions in the enterprise scope, several strategies have been proposed and some of them are partially implemented.

One is to develop a new, single, unified database for DataBase Management Systems (DBMS) and application programs to store and manage all the data types (Flatau, 1988). The concept of using a meta database to include both data and knowledge was proposed at the Rensselaer Polytechnic Institute (Hsu and Skevington, 1987; Hsu and Rattner, 1991). They developed Two-Stage Entity Relationship (TSER) modeling and feature-based integration methodologies. This approach requires a complete redesign of the database and application programs.

Another is to connect existing DBMSs and applications through a common interface. As one of the earliest manufacturing information systems, the Integrated Manufacturing Database Administration System (IMDAS) (Krishnamurthy *et al*., 1988) is such an example. They extended traditional database management technologies to integrate data processing functions, and created a common interface to facilitate the communications between application programs and IMDAS. Based on transactions of individual application systems, IMDAS retrieves and updates data through the common interface. The system partially connected databases and applications, but system level data integration remains unsolved. The difficulty lies in two aspects: the complexity and heterogeneity of engineering data structures and the long duration of engineering transactions.

A recent strategy is to develop Knowledge Based Systems (KBS) to support the interoperability between heterogeneous database systems. The framework proposed at the University of Waterloo (Dilts and Wu, 1991) uses KBSs to connect individual databases. They designed one KBS for each manufacturing application program, and all these KBSs are connected through a common intelligent interface. Queries from one application program are sent to the interface, which analyzes these queries and sends them to appropriate KBSs and DBMSs for execution, and then sends the integrated results back to the inquirer. By interactions of KBSs and the interface, the system intelligently retrieves and updates databases to realize data integration. Their approach realizes partial

system integration within a company, but system implementation details are not provided. Another framework proposed at the University of Maryland also uses a KBS to control information flow between application systems (Harhalakis *et al*., 1995). Their research focuses on the development of a knowledge base, which contains rules representing company policies. The system consists of a central KBS and a distributed database management system to manage the information flow between component databases, while issues for the distributed partners and knowledge are not addressed.

Of the above strategies, using KBS seems to be a promising way for manufacturing information integration. However, current approaches have only realized static connections of component systems within a company, while flexible and dynamic information integration among multiple firms in a virtual enterprise remains unaddressed. Our work attempts to fill this gap and provide a flexible yet consistent information framework for agile partnerships.

2.2 Collaborative Architecture Engineering Construction design

Collaborative design has been a research topic mainly in the Architecture Engineering and Construction (AEC) domain. Many prototype systems as well as mechanisms have been proposed to support collaborative design.

Current database systems in collaborative AEC design have two architectures: the central model and the distributed model. The central architecture has a virtual central database that is accessible by multiple partners. An example is the DICE system (Distributed and Integrated environment for Computer-aided Engineering) being developed at MIT (Sriram *et al*., 1991). DICE can be envisioned as a network of computers and users, where communication and coordination is achieved through a global database and a distributed control mechanism. The prototype includes: (i) Blackboard, which is a global object-oriented database and acts as the medium for user communication and cooperation, (ii) Knowledge Modules (KM) which contain on-line tools and libraries for designing and construction, and (iii) Control Mechanism (CM), which evaluates individual KM actions and assists in partner negotiation. These three parts cooperate with each other intelligently to provide a consistent and synchronized way for co-workers to perform collaborative AEC design. The distributed model includes distributed AEC databases where data dependencies are maintained through validating inter-database constraints (Tiwari and Howard, 1994). A Local Constraint Manager manages the portions of constraint specifications relevant to a local site, and a Global Constraint Manager maintains global design constraints. Of these two models, the central model puts more constraints on partner applications, while the distributed model is more complicated to implement.

Data sharing is a basic requirement in collaborative AEC design. To complement the functions of relational databases, which are good at relational data accessing and transaction management, and the object-oriented applications, which provide high data structure abstraction and flexible object manipulation, a front-end framework was built in Stanford University (Law *et al*., 1991). The framework consists of an object interface, which is built to connect the applications and the persistent storages, and functions as an object manager to create, delete and update objects and view objects. An essential issue in data sharing is data consistency maintenance. For central database systems, traditional techniques include Two-phase-locking, Time-stamping, etc. (Elmasri and Navathe, 1994), and modern approaches adopt strategies using lock management, version management, etc. (Sriram *et al*., 1992). For distributed database systems, data consistencies are enforced through the classic Two-phase-commit protocol, while new approaches include: (i) building a transparent interface between knowledge-based applications

and databases to maintain constraints (Howard and Rehak, 1985), (ii) decomposing global constraints into local constraints and validating global constraints through validating local constraints (Tiwari and Howard, 1994).

Current AEC design systems emphasize only the collaborative designs. However in agile manufacturing, approaches are needed to support agile partner collaborations not only in product design, but also in process planning and resource management. In AEC domain, building global databases and maintaining distributed database static constraints are the main approaches. Our work provides an integrated framework, which supports collaborative product development across domains and companies. Constraints among distributed partner databases are maintained dynamically, flexibly and intelligently.

2.3 System Development Tools

Many system design and modeling tools have been developed. The ICAM Definition (IDEF) proposed by the United Air Force was a primary tool for system development. IDEF includes IDEF0, IDEF1, IDEF2 and so on. IDEF0 models the system from the functional perspective, IDEF1 from the informational perspective, IDEF2 from the dynamic (system simulation) perspective. All together, IDEFx provides a complete system modeling method.

Petri Nets (PNs), on the other hand, were introduced to model concurrent, asynchronous, distributed, nondeterministic and stochastic systems (Peterson, 1981; Murata, 1989). In the manufacturing domain, PNs have been mainly used in modeling manufacturing cell controls (Boucher *et al*., 1990; Teng and Black, 1990). In manufacturing information system development, PN's powerful synthesis and validation techniques have been used in developing Knowledge Based Systems (Lin, 1991).

Using objects, Object Oriented Methodology (OOM) simulates system mechanisms and captures system entity attributes dynamically and accurately. OOM has been widely applied in manufacturing system modeling and analysis, examples include the object oriented FMS model (Adiga and Gadre, 1990), the unified modeling framework called Information Processing Object Hierarchy (IPOH) (Sun and Lin, 1992; Changchien *et al*, 1994), and the M*-OBJECT methodology in manufacturing information system development (Berio *et al.*, 1995). We will draw upon OOM concepts and use Petri Nets to model and validate our knowledge based information controllers.

3 Agile Manufacturing Information System Requirements

Agile manufacturing systems are proposed to support partner cooperations so as to penetrate the global market (Nagel and Dove, 1993). The operational unit in agile manufacturing industry is a dynamic virtual enterprise consisting of agile partners who collaborate with each other. The overall objective of an agile manufacturing industry is to achieve quick response to marketing demands, with compatible product quality and lower manufacturing cost. This section discusses the various requirements on the information and information transaction management in the agile collaborative product development process. The purpose is to analyze the system requirements needed to support collaborative activities in product design, process planning and manufacturing resource planning, so as to design and implement the information systems supporting these requirements. Our goal in this effort is to least disrupt partner companies' established procedures for design/manufacturing, and to encourage cooperative virtual manufacturing. In this view, to support collaborations among product designers, process planners and

manufacturing resource planners at various stages of product development, AMIS must possess the following functionality.

1. **Provide consistent information to distributed partners** Partners are distributed geographically and use data replications or local databases. Inconsistent data at different partner sites will lead to product misdescription or even manufacturing chaos. It is up to the AMIS to make sure that every transaction brings the agile manufacturing system from one consistent state to another. To maintain the system information consistency, AMIS is required to support the following.

   (a) *Representation and maintenance of data dependencies across companies.* Different from traditional manufacturing enterprises, where data is sitting in a single company under the same information management policies, a virtual enterprise has distributed partner database systems across different companies. Data dependencies exist between different partner database systems. AMIS must specify and maintain these inter-company data dependencies.

   (b) *System transaction atomicity assurance across companies.* Information in a virtual enterprise is distributed at various partner sites. Transactions usually involve multiple partners. It is critical to make sure that every partner transaction brings the whole system from one consistent state to another, which means that every transaction must process atomicity: either be committed as a whole, or not be committed at all.

2. **Resolve multiple data ownership and form multiple information views** Agile partners are from various companies and perform diverse tasks in the product development process. On one hand, to support partner collaborations across companies, multiple information views need to be built from the same primitive data storage, and the same primitive information storage may be updated from different partner views. On the other hand, multiple data ownership exists in a virtual enterprise, which limits the primitive data updating. AMIS must resolve the inter-company data ownership by mutual agreements to form multiple object views.

3. **Reflect partner individual policies and retain partner autonomy** In traditional manufacturing enterprises, there exists a single set of policies within the company's scope. In a virtual enterprise which consists of multiple partner policy sets, AMIS must reflect the partner's own policy set regarding the information flow in/out of that partner. To join the agile manufacturing network, it is usually a big concern that individual partners keep their policies independent, while mutual cooperations can be guided through mutually agreed protocols. So it is necessary for the Agile Manufacturing Information System to provide flexible schemes to reflect company specific policies and procedures.

   An agile partner in a virtual enterprise is dependent and independent. It is dependent in the sense that it cooperates with others to develop products; it is independent in the sense that it has full autonomies in executing its policies. Flexible schemes must be provided so that partners' heterogeneity and autonomy are unaffected in a virtual agile enterprise. AMIS needs to provide the following.

   (a) *An open system architecture.* A manufacturer should be independent in joining the agile manufacturing network or leaving a virtual enterprise. As an analogy, our ultimate objective is to provide a "phone

jack", and it is up to the manufacturer to decide whether to "plug in the phone and connect to the network" or not. As the first step towards this goal, AMIS needs to have an open system architecture.

(b) *Partner policy independency.* After joining a virtual enterprise, a partner needs to retain its autonomy. AMIS must provide flexible partner cooperation schemes so that individual partner policies are reflected and partner autonomies are retained.

4. **Accommodate partner heterogeneity** Heterogeneity in partner computer hardwares, database systems and applications is the reality in virtual enterprises. To achieve partner information sharing and concurrent transaction management. AMIS must accommodate partner's heterogeneity.

5. **Provide necessary security to partners** Cooperations between agile partners are agile and dynamic. Necessary procedures must be provided to ensure individual partner information securities.

   (a) *Information access control.* Information access from outside partners to a local manufacturer should be controlled by the local manufacturer's site, through procedures such as user authentication, group authentication, firewall setup, restricted access, etc. We only use user authentication in our system implementation, and hope that more robust access control techniques can be adopted in more sophisticated implementations.

   (b) *Secure information transmission.* Advanced technologies should be provided so that messages between partners are kept secure. Current approaches include cryptography techniques, etc. We will leave this issue to the data transmission domain, and narrow our focus on the other system requirements.

In this section, we have described the information presentation and manipulation requirements in agile collaborative environment. We will develop our information system model based on these requirements. Nevertheless, this is not a complete list of AMIS requirements, there are many other requirements such as teleconference supporting, partner concurrent collaboration supporting, etc. We focus on developing a framework which provides partner database interoperability, maintains data consistency between partners, reflects partner individual policies and provides multiple object views, while accommodating partner heterogeneity and retaining partner autonomy. In the following sections, we propose an integrated information framework which satisfies these requirements.

It has been brought to our attention that the information security issue is extremely important in setting up the agile manufacturing network. However, this is beyond our current effort. It is our hope that in the future, other research will address this issue and provide advanced technologies to secure information accessing and transmission, and altogether, provide a secure and flexible agile manufacturing network. At this time it is hard to anticipate any reasons why our framework will not be able to work in conjunction with security supporting technologies.

4 Overview of the Virtual Information System for Agile Manufacturing

In order to satisfy the requirements of an AMIS detailed in the previous section, we overview here our proposed framework – the virtual information system for agile manufacturing. While alternative choices may exist, our

framework is guided by the concept of the virtual enterprise structure composed of several physically distributed manufacturing companies. As a parallel to this structure, we propose a distributed virtual information system composed of several partner information systems. Further, our framework is based on the fact that each partner is autonomous and has specific policies on information management. First, we present the virtual enterprise and information system architectures. In the second part, we discuss a single partner unit of the virtual information system, the Partner UNit Information System (PUNIS) which resides at a specific partner site and will remain our main focus in this paper. Finally, we overview the functions of these information units in the virtual system with the help of an example scenario. In summary, we present a virtual enterprise information system model from three aspects: the virtual enterprise information system architecture, the partner unit information model and the enterprise information operational model.

4.1 Virtual Enterprise and Information System Architecture

Considering the cooperative business practices and the emerging agile environment, heterarchical structures are appropriate for virtual enterprises. Figure 1 gives the snapshot of a virtual agile enterprise and the individual agile companies – the partners in the virtual enterprise.

As shown in Figure 1, the virtual enterprise is composed of three autonomous partners: company A, company B and company C. The network connects them and provides channels for their collaborations. Company A, B and C have agreed upon being partners and working cooperatively on the development of a new product P. The profile for the enterprise is: company B is specialized in design and is responsible for product Computer Aided Design (CAD); company A has experts in Computer Aided Process Planning (CAPP), and is chosen as a partner to develop process plans. Company C is to construct enterprise Manufacturing Resource Plans (MRP). This heterarchical agile enterprise has the following characteristics: (1) *The system is open and modifiable.* A partner can be connected to or disconnected from a joint-venture virtual enterprise. Partner selection is the process of partner mutual interaction and connection establishment. (2) *The working partners are fully autonomous.* In this example, companies A, B and C are partners, they have full autonomies, and they cooperate with each other through mutually agreed protocols.

On the other hand, the virtual information system has the following features:

1. Each partner is connected to the network through a Client and a Server. The server is responsible for serving other partners. It receives queries, interacts with the DataBase Management Systems (DBMS) to execute the queries and sends the results back to the inquirer. The client is responsible for coordinating the workflow execution. It sends subqueries to other partners and receives results.
2. The local Workflow Manager controls the information flow. Based on the query from the user, the workflow manager has to retrieve relevant rules to analyze the query, and form workflows to be executed by the client to keep system data consistency.
3. The correct workflow execution is ensured by the cooperations of partner servers and clients. The client at the global workflow initialization site becomes the temporary workflow coordinator. It uses dynamic two-phase-commit protocol to ensure the workflow execution atomicity.

Figure 1 around here

Now zooming into a single partner, as shown in Figure 1, there exist seven modules in a typical agile partner model: the local primitive information storage (database), the DataBase Management System (DBMS), the Knowledge Base, the Workflow Manager (WM), the engineering applications, the partner client and the partner server. Local primitive information storage stores local primitive information and forms a part of the enterprise primitive information storage. DBMS manages local primitive information accessing, updating and constraint maintenance. By interacting with the DBMS, the server is responsible for local atomic object generation and instantiation. Application programs access information storage through the Workflow Manager (WM), the Knowledge Base stores partner policies and inter-partner protocols supporting partner transaction workflow specifications. The Client and Server are responsible for connecting to the network and transmitting information back and forth. The partner unit information system has the following features:

1. With the manufacturing applications (CAD/CAPP/MRP), DBMS and local database storage, each site is a stand-alone system and has its autonomy.
2. Adding the software modules such as the Server, the Client, the Workflow Manager and the Knowledge Base, the partner becomes an active node in the agile manufacturing network. With the server connected to the local DBMS, the partner makes local data accessible to the other partners. With the client, the partner may access the other partners' data.
3. The workflow manager is responsible for converting a partner query to a partner workflow. A workflow is defined as a collection of subqueries (single-database queries) organized to fulfill a business task. The workflow manager consults with the knowledge base to decide whether the partner query is local transaction only (i.e., transactions that involve only local data) or involves global transactions (i.e., transactions that access global data). This brings flexibility, with some trade off on the efficiency.

In summary, our architecture emphasizes global integration while keeping the autonomy, heterogeneity and geographical distribution of agile partners, and maintaining the dynamic property of the virtual agile enterprise. Within such an architecture, agile partners are able to collaborate flexibly and dynamically to develop high quality products with quick response to customers.

4.2 A Partner Unit Informational System Model

The partner unit information model describes the information representation method and control mechanisms of a single partner of the virtual enterprise. We use two parallel hierarchies to represent the model: the data hierarchy and the transaction workflow hierarchy. These two hierarchies form two sets: (1) the set of partner information objects (i.e., atomic objects constructed at a single partner site and composite objects formed from multiple partner sites), and (2) the set of transaction workflows to form and manage partner information objects. These two sets are mutually related to each other. We use a knowledge base to store the partner information management guidelines: the partner policies and inter-partner protocols.

As shown in Figure 2, the data hierarchy represents the information in an object oriented fashion. At the bottom of the hierarchy is the local primitive information storage needed to represent basic product and enterprise resource data. The next to bottom layer contains Atomic Objects, which are usually created by local partners within local domains. Objects combined from local and/or remote domains compose the upper layer, the Composite Object

layer. From the enterprise point of view, these three layers (aggregated over each partner) compose the enterprise data hierarchy. The data hierarchy is further discussed in Section 5.1.

The workflow hierarchy, on the other hand, specifies the partner information transactions. Corresponding to the structure of the information hierarchy, at the bottom of the workflow hierarchy is the set of primitive data transactions. The upper layers consist of the local workflows and global workflows. The workflow hierarchy is elaborated further in Section 5.2.

The Knowledge Base, composed from local partner policies and inter-partner protocols, is used to compile the workflow specifications and manage partner information flows. The knowledge base consists of expert rules controlling partner information exchanges. It is briefly discussed in Section 6.

Using these two hierarchies and the knowledge base, we are able to represent and manage information for a partner, and subsequently in a virtual enterprise, to support partner collaborations. Our main task, then, is to form these two hierarchies, and the knowledge base which controls the information flow in a consistent manner.

Figure 2 around here

4.3 Operational Model: A Sample Scenario

To describe the information flow within a virtual enterprise, we use a sample scenario to illustrate its operational model: the collaboration between an MRP planer and the design engineers in the virtual enterprise.

As shown in Figure 3, (only modules related to our sample process are shown in the figure), company Y and Z form a virtual enterprise to design and manufacture product P. Suppose designers D1 and D2 from company Y are working on the design of P. D1 and D2 share various design objects, which are generated from primitive information storage at site Y. At a certain time, an MRP planner M from company Z tries to work on P's MRP report. M needs P's Bill-Of-Materials (BOM) information. M sends a query to the local workflow manager through the MRP application program, which interacts with the local knowledge base and compiles a transaction workflow. The local client then sends this BOM request to company Y. The server at site Y, after receiving the query and performing identification checks, interacts with the DBMS in site Y to generate and instantiate a BOM object from the primitive information storage. It then sends it to M and records M's identification and query in log files. Later on, when a new design version of P is reached by D1 and D2, and the primitive information storage is updated, the WM at site Y consults with local knowledge base and log files, compiles workflows to be executed to keep the global data consistency.

From this sample process, we can see that the cooperation between partners is realized through the cooperations of the partner clients and servers, and local workflow manager controls local information flow. As the main parts of the PUNIS, the partner WM and Knowledge Base reflect local partner's own policies and manage the information exchange between the local host and the outside world. As specified in Section 3, concurrent partner collaboration is beyond the scope of the current research.

Figure 3 around here

In summary, the VISAM information system for an agile manufacturing enterprise is virtual and formed by distributed partner information hierarchies, and the enterprise transaction workflow hierarchy is really the

aggregation of partner unit information transaction management systems. Information exchange among partners is managed by interactions among distributed partner WMs. In subsequent sections, we will look into issues about the data hierarchy, the workflow hierarchy and the knowledge base development in more detail, and present the prototype system which realizes the framework for illustrative purposes.

5 Data and Workflow Hierarchies

This section describes the two hierarchies in our framework: the data hierarchy and the workflow hierarchy. The data hierarchy represents the agile manufacturing information in a flexible, object-oriented, hierarchical structure, so that the agile partners can dynamically share the information from different views and at different abstraction levels. The basic idea is to provide Dynamic Object Views from primitive information storage. Focusing on the object framework development, we emphasize on the hierarchical structure as well as the methods to form the data hierarchy. Based on the three layer hierarchical structure discussed in Section 4, Section 5.1.1 describes the primitive information storage in agile manufacturing. Section 5.1.2 discusses the agile manufacturing objects, and presents the overall data hierarchy structure. In Section 5.2, we discuss the workflow hierarchy. Based on the discussion of these two hierarchies, Section 5.3 gives a detailed example with a sample data hierarchy, and a simple workflow used to form the composite object.

5.1 The Data Hierarchy

There are three layers in the data hierarchy: (1) the primitive information storage, (2) the atomic object layer, and (3) the composite object layer.

5.1.1 Primitive Information Storage

The primitive information storage in an agile manufacturing system provides necessary ingredients to form information objects used/processed by collaborative partners and applications. It includes fundamental data needed in agile design/manufacturing. In an agile manufacturing environment, the following major information types are needed as the primitive storage in the process of product development and enterprise resource planning. This information list is not exhaustive, and it may also differ depending on the specific industry sector.

1. Primitive Product Information. Product information basically describes what will be produced in terms of engineering drawings, product assembly structure and part lists, group codes and CAD databases, etc. Starting from conceptual specifications from customers, product information are enriched by designers and manufacturers throughout the product lifetime. Primitive product information are the fundamental data and knowledge describing product properties. It is the minimum information set for a product description. For example, the primitive information for discrete mechanical parts will include the following: (1) primitive geometrical information, (2) primitive topological information, (3) materials, (4) surface finish information, and (5) tolerance information (Qiao *et al.*, 1993). These information are primitive in the sense that they must be specified to represent the product, and that they are used to construct design/manufacturing objects throughout the product development process. From engineering design, feasibility and manufacturability

evaluations, to process planning and manufacturing resource planning, exchange of product data is essential for agile partners to keep track of product information so as to accomplish the whole production realization procedure consistently.

2. Primitive Process Information. Process data specifies the procedures to transform raw materials to final products in terms of manufacturing operations and activities, and possibly machine operation programs such as numerically controlled (NC) programs. In an agile manufacturing environment, primitive process information is defined as the minimum set of processes for making the product. It includes the primitive operations on the products and it relates to product design information and manufacturing resource information. Examples of the primitive process operations for mechanical parts will include primary processes (casting, injection molding, etc.), secondary processes (turning, drilling, etc.), and tertiary processes (grinding, etc.).

3. Primitive Manufacturing Resource Information. Manufacturing resource information describes production means in terms of manufacturing facilities (including machines, humans as well as applications) and their spatial, temporal and organizational properties. Manufacturing resource information are primitive in the sense that they are essential for partner selection, product design, design manufacturability evaluation, product fabrication and enterprise resource planning. In developing and manufacturing the product, manufacturing resources are the primitive means to generate the physical product. An example of the primitive manufacturing information is the data structure of a workcenter. The basic properties of a workcenter includes the following: (1) work center ID, a unique identifying number assigned to each work center in the system, (2) work center description, (3) department, (4) work center status (hold, working or released), (5) work center state (available or busy), (6) capacity parameters such as horse power, speed range, feed range, work envelope, accuracy, tool change time, feed change time, speed change time, etc.

Sitting at the bottom of the data hierarchy, the aforementioned primitive information are stored in various partner design/manufacturing databases at distributed partner sites, managed by various DataBase Management Systems (DBMS), and are used by high level partners and partner applications to form objects in the product development process. The data hierarchy is built from the primitive information dynamically, using the partner workflows, either locally or globally.

Current engineering information systems often lack the link between the primitive information storage and high level application objects (Law *et al*, 1991). Our approach is to construct a channel, which connects remote partner primitive information storage with partner application objects, and provides a common mechanism for distributed partners to access remote primitive information storage concurrently and consistently. We will discuss the approach in the following sections.

5.1.2 Agile Manufacturing Objects

Viewing the world as a collection of objects, Object Oriented Methodology (OOM) takes everything as objects (Rumbaugh *et al.*, 1991). For instance, a data value is an object, a manufacturing part is an object, an assembled product is an object, too. OOM provides a unified information view and flexible approaches for data exchange and

transfer; thus, it is appropriate for agile manufacturing information representation. Here we give a brief definition first, and then focus on the object generation and construction schemes.

(1) Definition

In agile manufacturing, we define an agile object as the following:

Agile Object(AO)=(ClassID:CID, ObjectID:OID, Constructor:C, Attributes:A, Methods:M)

where

CID is the identification of the class which AO belongs to;

OID is the unique object identification;

C is the function specifying the procedures for generating the object;

A is a finite set of object attributes, including both static and dynamic attributes;

M is the set of methods to access and pass messages to the object.

Here, we are interested in the constructor C as we need to build the scheme to construct objects. We divide agile objects into two categories: atomic objects and composite objects. Atomic objects are defined and constructed directly from primitive information storage at a single partner site, while composite objects are constructed from atomic or composite objects, which are formed at different partner sites. The atomic objects are domain specific and need to be coded by individual partners or generated by advanced applications during the product development process. For example, in designing a mechanical assembly P, the constructor for the object "BOM of P" is the function to form the BOM of product P, it can be specified by the assembly designer, or be generated automatically using applications in the design process. Composite objects, on the other hand, are constructed from objects, and the partner needs to specify algorithms for constructing the object. In short, it is the partner's responsibility to give the object constructor, and given the constructor, the information system integrates the distributed engineering databases and forms the object automatically.

(2) Object Construction

As we have discussed, atomic objects are formed directly from primitive information and are usually formed by a local partner from the local database storage. On the other hand, composite objects are constructed from atomic objects or composite objects, which may not be located at a single site. So we are mainly interested in the composite objects. An example of a composite object is the product assembly which is constructed from object o1 at partner site P1 and o2 at P2. To form composite objects, it is essential for remote partners to process the subqueries and pass atomic/composite objects between distributed partner sites.

To construct a composite object, we need to specify the location and formation algorithms for the sub-objects, the dependencies between these sub-objects, and the algorithm for forming the object from these sub-objects. We have developed the Partner Query Language (PQL) and the Partner Workflow Language (PWL) to specify the object construction and manipulation (information transaction) queries (refer to Section 5.2.1 and 5.2.2 for more details of PQL and PWL).

(3) The Data Hierarchy

From the above discussions, we can see the enterprise data hierarchy aggregates the distributed partner databases into an integrated virtual database. These distributed partner manufacturing databases are not simply connected

physically. These databases are connected and managed by a certain scheme so that they satisfy the agile manufacturing information system requirements as a whole unified database. The database hierarchy has the following properties:

1. Flexible and dynamic object construction and sharing schemes. The virtual enterprise data hierarchy is composed of individual partner information hierarchies, which aggregate together to form a complete enterprise information representation. The primitive information is stored at distributed partner sites, the atomic objects are constructed from local storage, and the composite objects are formed by using other atomic objects and composite objects located at local or global sites. As shown in Figure 2, there are three layers in each partner site: the bottom is primitive information storage, the middle is atomic objects, and the top is composite objects. A composite object in partner A can be formed from the atomic objects from partner A, B and C, and the composite objects from Partner A, B and C.
2. Multiple object views. Based on the distributed primitive manufacturing information, atomic objects and composite objects are built from different partner views. For example, a product is an assembly of engineering drawing and specifications from the product designer's point of view, it is an aggregation of primitive processes from the process planner's perspective, and for an MRP user, a product concerns the work schedule, the inventory and the purchase orders. Because of the partner information exchangeability, our framework allows the formation of multiple object views from the same primitive storage. Recently, EXPRESS-V and EXPRESS-X are also being developed as standards for this (Spooner and Hardwick, 1996).

In the next section, we cover the schemes for constructing and manipulating these objects: the workflow specification and execution schemes.

5.2 The Workflow Hierarchy

Based on the enterprise data hierarchy, we divide partner transactions into two categories: the local transactions and the global transactions. Local transactions are those which involve local atomic objects and primitive storage and are described by local workflows. Global transactions are transactions which need to access or manipulate data located at multiple partner sites and are represented by global workflows.

The workflow hierarchy consists of workflows managing partner object constructions and manipulations. Since our focus is the workflow specification and execution schemes, we develop languages for partners to specify object construction and manipulation queries, and operational scenarios and protocols for partner query executions.

5.2.1 Partner Query Language

Partner Query Language (PQL) is developed to facilitate partners to compile partner queries to the distributed manufacturing databases. From a partner's point of view, when the information is logically stored in a single site, the Structured Query Language (SQL) for relational databases (Elmasri and Navathe, 1994) is appropriate for compiling the query. We adopt the SQL format in our Partner Query Language. The basic syntax for a partner query is the following:

construct (or create, delete, update) Object_name
from partner_database_schema
where condition_clause

The object_name is the template of the object(s) to which the query is applied. The partner_database_schema is the set of schema in which the data or objects reside. The condition_clause specifies the conditions needed in the query. An example is to construct a composite object from atomic objects at distributed partner sites. Assume in our previous sample enterprise (Figure 1) formed by partner A, B and C, the enterprise Inventory information is stored in an MRP database, the product Gross Requirement (GR) is stored in the SALES database in the sales department, and the product data is stored in a CAD database, then the query for the MRP partner to construct the object "Production Order Release (POR)" for part #32 would be as shown in Figure 4.

To compile partner queries using PQL, the sharing common data schema between partners must be built according to mutual agreements. For existing databases, inconsistencies in database tables and attributes can be resolved using the heterogeneous database discrepancy resolution (Rusinkiewicz *et al.*, 1995). PQL ensures the partner query's integrity by using the two-phase-commit protocol in executing its corresponding workflow.

5.2.2 Partner Workflow Language

Partner queries compiled using PQL do not specify the query details such as partner sites and object construct algorithms. The system is responsible to find out where these objects (BOM, GR and INVENTORY) should be formed and transformed (see also Section 5.2.3 for mapping from PQL to PWL). Since the partner query is going to be executed as workflows, we developed the Partner Workflow Language (PWL) to specify partner global and local transaction details. Here we describe this language by using a sample query and giving its syntax and semantic meanings (see Song, 1996, for further details).

A sample query written in PWL is shown in Figure 4. This query is actually based on the partner query shown on the left side. The syntax of a partner workflow query is the following.

beginprogram
query_components
endprogram

Figure 4 around here

The **query_components** contain four fundamental components: objects, subtask definitions, subtask dependencies and preference specifications.

(1) Objects

The objects here are a broad concept: they can be atomic objects or composite objects in the data hierarchy (the usual case), they can also be the primitive information storage. Objects are used in PWL as arguments or results of the partner subtasks.

The type of an object specifies its categories. The object type is basically a user-defined object structure. It can be the basic data types such as "Integer", "CharString", etc., if the object is actually the primitive data storage.

It can be an aggregate set of basic types, if the object is an atomic object. It can also be an aggregate set of objects, if the object is a composite object. For example, the object BOM has the following type:

object BOM of	
Part_Id:	Integer
Part_name:	CharString
Child_Id:	Integer
ChildQuantity:	Integer
endobject	

Passing objects through the network is an active research topic (Mowbray and Zahavi, 1995). Currently, we focus on the object data passing, and in our system implementation, we use a certain format to represent object data. For example, data for "BOM" is represented using the format of "parent-quantity (child-quantity)" with a recursive format. Particularly, the BOM for part #32 is:

"32-1(321-2(3211-1) 3211-1)" (which is recursive parent-#(child-#) format)

In this manner a BOM is encoded at the sender site and decoded at the receiver site.

(2) Subtask definition

Subtasks are the partner subtransactions executed at individual partner sites. A subtask is defined by the following specifications:

i. The subtask name,
ii. The input object specified by the name of the subtasks generating the object,
iii. The partner host site, specified by the domain name of the host IP address,
iv. The partner software system name,
v. The valid subtask execution time period,
vi. The subtask execution procedure,
vii. The commit option when the subtransaction is instructed to commit,
viii. The undo option when the subtransaction is instructed to abort,

(3) Subtask dependency specifications

This part specifies the control flow or the subtask execution order of the global workflow. It actually synchronizes the subtask execution order. The subtask dependency specification is based on the data flow and control flow of the global transaction. Generally, subtask dependencies include subtask ordering dependencies defining subtask execution orders, and trigger dependencies defining contingency executions (Zhang, 1994). The dependency specifications in the sample workflow (shown in Figure 4) are the following:

i. subtask t1, t2 and t3 can be executed in parallel,
ii. only subtask t4 can be executed only after t1, t2 and t3 succeed,
iii. if t4 succeeds, the global workflow succeeds,
iv. if t1 or t2 or t3 fail, the global workflow aborts.

(4) Alternative workflow specification

Alternative workflow specifies the workflow which is an alternative to "this" workflow, while workflow preference defines a certain criteria for committing/aborting the subtasks.

5.2.3 Mapping from PQL to PWL

We have discussed the Partner Query Language (PQL), which is used by individual partners and reflects the logical view of an aggregated database, and the Partner Workflow Language (PWL), which is constructed by the system and reflects the logical view of relationships between distributed partner database systems. The mapping from PQL to PWL is performed by consulting the system data dictionary, and the knowledge base consisting of partner information sharing policies. Considering the partner query and the workflow query in Figure 4, the mapping from a partner query to a workflow query is performed by the following steps:

i. Based on the component database systems specified in the PQL, find out the relevant partner IP addresses and software systems;

ii. Based on the component database tasks specified in the PQL, construct the workflow subtask specifications. This is accomplished by checking the global data dictionary;

iii. Based on the transaction specific expert rules stored in the knowledge base, and the component database dependencies, build the subtask dependencies including execution order, trigger and real-time subtask dependencies;

iv. Specify any alternative workflows or subtask preferences.

In our framework, the Workflow Manager (WM) is responsible for constructing PWL queries from PQL queries. To perform the mapping steps, WM needs to consult the expert rules stored in the knowledge base. Basically, the knowledge base contains two portions of information: one is the partner information regarding the data schema, the host address and the software systems of relevant partners; the other is transaction specific rules specifying partner information transaction management policies (Section 6).

5.2.4 The Workflow Execution Scheme

PWL specifies the locations and dependencies of the distributed partner data needed to execute the query, and how to use the information. Actual mechanisms are needed to implement the specified algorithm. Here we describe the partner workflow execution protocols first, and then give a scenario which realizes this implementation by two-way communications between partner database systems.

(1) Partner workflow execution protocol

The whole purpose for building the data hierarchy and the workflow hierarchy is to represent the data in an object oriented form and to let the partners share the information across the enterprise dynamically. Based on the information system requirements presented in Section 3, we use the following partner workflow execution protocols as guidelines.

(1) Dynamic Two-Phase-Commit protocol. This protocol is based on the traditional "Two-Phase-Commit" Protocol. It is dynamic in the sense that the system coordinator role is no more central, the role is dynamically assigned to the client at the workflow initialization site, which may be any one of the multiple partners in the agile enterprise. To execute a partner query, the workflow manager decomposes it into subqueries (subtasks) and specifies their relationships. The local client (temporary system coordinator) sends out all the subqueries and a vote message to each participant partner server. Each partner server enters into the preparation state for subquery execution and responds with a vote. After the client collects all the votes, it makes a decision based on the votes, and broadcasts the decision to all the participants. If all positive votes are received, it sends "commit" commands to all component partner servers and waits for results; otherwise it aborts the whole workflow by sending an "abort" command. This protocol guarantees the workflow execution atomicity. In the agile manufacturing environment, a central system coordinator is inappropriate in the virtual enterprise structure; thus the client which tries to execute the workflow becomes the temporary system coordinator. The workflow may or may not succeed, depending on the partner local system states.

(2) Partner data ownership protocol. This protocol requires that all data updating transactions must be performed by the data owner. Negotiation may exist, but final data updating is made by the local partner system. In an agile manufacturing environment, this is essential because of the existence of multiple data ownership. For example, update of the BOM object can only be conducted by the CAD user.

(3) Inter-partner data dependency reservation protocol. This protocol requires that each transaction must not violate the inter-partner data dependencies and constraints. Particularly, if local partner data is to be updated, the local workflow manager and client are responsible for compiling compensation workflows to update relevant objects. For instance, when performing BOM updating, the CAD site is responsible for compiling workflows to change the status of relevant parts to "hold" at the other sites.

These three protocols are mutually consistent. The Dynamic Two-Phase-Commit protocol makes sure that each workflow brings the system into a data consistent state, while the partner who owns the data is responsible for local data updating: whenever a local updating is to be made, corresponding compensation workflows need to be compiled and executed according to the data ownership protocol, and if a workflow is executed, it is executed completely or not executed at all.

(2) Partner workflow execution schemes

With the partner workflow execution protocols in mind, a workflow execution scheme is developed. The idea is to build a server and a client at each partner site, which implements the necessary functions. The complete mechanism is as the following.

1. At local site, a local server (e.g., Unix daemon) is launched and it listens to the network all the time so that the local partner is ready to provide service.
2. At local site, if the partner compiles a partner query, the workflow manager consults with the knowledge base and converts the partner query into a partner workflow which consists of a set of subqueries.

3. At local site, the client program is invoked to act as the temporary system coordinator. It tries to connect to the partner database systems at various partner sites by sending them the subqueries and votes.
4. At another remote partner site, just like the local site, a server has been launched to listen to the network all the time. When the server receives the connection request from the client, it performs authorization checks, and sends back a positive vote when the system is ready.
5. At local site, if the client receives all the positive votes, it issues the "commit" command to relevant partner servers, else it issues the "abort" command.
6. At the remote site, if a "commit" command is received, the remote server (daemon) then forks a child process, which works with the remote database management systems to execute the subquery. At the same time, the parent process listens to the network for any other possible connections. If an "abort" command is received, the remote server disconnects from the local client, and continues to listen to the network.
7. At the remote site, the child process sends the subquery results back to the client, and records the partner ID, the transaction time and subquery in the log file. Later on, if local information storage is updated, a new workflow is formed to look up all the logs and notify relevant partners to keep information consistent.
8. At local site, after the client has received all the necessary information from all the partners, it commits the workflow completely. Of course, all the information are recorded by the local logs for future tracking.

5.3 A Detailed Example

5.3.1 A Sample Data Hierarchy

In this part, we give a simple data hierarchy example. As discussed in Section 4, the sample virtual enterprise has three partners: company B is responsible for design, company C for manufacturing resource planning and A for process planning. Applying the above agile object concepts in our sample virtual enterprise, a simple data hierarchy can be built. Figure 5 shows a simplified data hierarchy and its layers.

1. Figure 5a shows the different layers of the data hierarchy. Figure 5b is the primitive storage layer. Primitive product data is stored at partner B, and the primitive resource data is stored at partner C. Figure 5c is the atomic object layer. Atomic object BOM is formed at partner site B, atomic object INVENTORY is formed at site C, object GR (Gross Requirement) is formed at the sales department (not shown in the figure). Figure 5d is the composite object layer. Composite object Production Order Release (POR) is formed based on BOM, GR and INVENTORY.
2. The primitive information storage layer includes two parts: i) the primitive product data stored at site B (schema 1), where table 1 specifies the structure of part #32, and table 2 specifies the structure of part #321. ii) The primitive inventory information stored at site C (schema 2), where table 3 stores the current inventory for part #32, #321 and #3211.
3. The atomic object layer is formed using the primitive storage. BOM for part #32 is formed from table 1 and 2 and is shown in Figure 5c.

4. The composite object layer. From the part Gross Requirement (GR) passed from the sales department and the BOM object from company B, based on the atomic object INVENTORY, we can construct the composite object "POR for part #32". Assuming the leadtime and lotsize for part #32, #321 and #3211 are the following.

The algorithm POR_FORMATION has two parts.

(1) Constructing POR for a particular part.

For a particular part, in period $t = [0, T]$, given initial inventory $I(0)$, Gross Requirement $GR(t)$, lead time 1, lotsize s ($s = 0$ means lot-for-lot), POR is formed as:

For time period $t = 0$ to T do $\{$
if $I(t) > GR(t)$ then $I(t+1) = I(t) - GR(t)$;
else if$(s \neq 0)$ then $POR(t-1) = s$; $I(t+1) = s + I(t) - GR(t)$;
else $POR(t-1) = GR(t) - I(t)$; $I(t+1) = 0$; $\}$

(2) Calculating GR for child part from parent part.

Given the parent (ID is parentID) part's BOM as an array Q(parentID)(childID), the child (ID is childID) part's GR(t) is formed as the following.

Search the BOM tree to find all the parents
For each parent, which has $POR(t)$
For time period $t = 0$ to T do $\{$
$GR(t) += Q$(parentID)(childID) $\times POR(t)$; $\}$

Figure 5 around here

5.3.2 A Sample Workflow Specification and Execution

In the simple data hierarchy presented in Section 5.3.1, the composite object POR is formed from three atomic objects: GR, BOM and INVENTORY. Examining the sample transaction "Build the Production Order Release (POR) from GR, BOM and INVENTORY," the following steps are employed to specify and execute the workflow.

1. Partner constructs the query using PQL, the query is shown in Figure 4;
2. Workflow Manager retrieves relevant rules from the knowledge base and forms the workflow using PWL, the workflow is shown in Figure 4 and is composed of the following subqueries:

   subquery1: Partner B construct BOM object for part #32;

   subquery2: Partner C retrieves local INVENTORY object;

   subquery3: Partner C reads GR from sales department;

   subquery4: Partner C constructs the POR object using the received objects;
3. Local client is invoked as the system coordinator, which executes the workflow based on the above specifications. The executing process is shown in Figure 6.

Figure 6 around here

5.4 Summary

In this Section, we have discussed the agile manufacturing object concepts and construction schemes. We have discussed partner workflow specifications and execution schemes. The information hierarchy represents enterprise information with multiple object views. The partner workflows build and mange these objects. We have also introduced the dynamic two-phase-commit protocol to keep the transaction atomicity. Partner data ownership and partner data dependency mechanisms are introduced to keep the system data consistent.

Per our discussion, the partner primitive information builds the ground of the information hierarchy. Using object orientation, various atomic objects and composite objects are built using partner workflows. Currently, data objects are encoded using a certain data format (e.g., recursive parent-#(child-#) for BOM object). Such encodings can be replaced in the future by the emerging CORBA standard (Mowbray and Zahavi, 1995). The workflow hierarchy is used to build and manage the information hierarchy. These two hierarchies are mutually related and provide consistent, dynamic and multiple views to distributed collaborative partners.

6 The Knowledge Base

The knowledge base stores expert rules reflecting partner information management policies. By interacting with the knowledge base, the workflow manger is able to convert partner queries into workflow queries, and manage the local client to execute the workflow in a consistent way. Building the knowledge base is a critical step in our framework development. In this section, we present an approach for building the knowledge base. Our approach includes the following steps. The details are described in subsequent sections (see also Song, 1996).

1. Knowledge acquisition. Form the set of partner generic policies and inter partner protocols for partner information exchanges. These abstract policies are further decomposed into subtransaction specifications using a "Top-down" approach (see Section 6.1).
2. Knowledge modeling. Use Petri Nets (PN) to model the subtransactions (see Section 6.1). The purpose is to use PN verification techniques to assure knowledge consistency.
3. Petri Net synthesis and verification. Bottom up, starting from subtransactions, synthesize low level PNs into higher level compound PNs, then use PN verification techniques to verify all levels of PNs, so that inconsistencies and redundancies in these PNs are eliminated (see Section 6.2).
4. Knowledge base formulation. Build expert rules to specify partner policies and inter-partner protocols based on the Petri Nets models (see Section 6.3).
5. Knowledge base validation. Testing the knowledge base with the sample transaction management policies.

6.1 Knowledge Acquisition and Modeling

The knowledge base is designed for and used to reflect partner policies and protocols to control the information flow between agile partner applications. The rules embedded in the knowledge base specify the constraints and relations between the distributed manufacturing database systems located at various partner sites. The necessary knowledge includes two parts: (1) knowledge about virtual enterprise partners including: (a) Partner user ID and

password, host IP address; (b) Partner database schema; (c) Partner data dependencies, and (2) knowledge about partner information control policies including: (a) local partner policies governing manufacturing transactions; (b) inter-partner information exchange protocols.

The first part can be built in the process of partner interaction and virtual enterprise formation. The second part are generic policies and can be formed by individual interviews and group meetings with enterprise partners. We assume the first part is available, and focus on the second part. For simplicity, our prototype system rules are extracted from our own industrial experience and from Lin (1991).

Generally speaking, any "partner policy" and "partner protocols" start from abstract specification about the general global rules which control transactions. Sample transactions in AMIS include the following.

Product data (CAD) transactions	Process Planning (CAPP) transactions
(1) Add new product parts in CAD;	(7) Building new process plans in CAPP;
(2) Add new component relationships in CAD;	(8) Deleting process plans in CAPP
(3) Deleting product parts in CAD;	(9) Modifying process plans in CAPP;
(4) Deleting component relationships in CAD;	Resource Planning (MRP) transactions
(5) Revising product parts in CAD;	(10) Deleting work centers in MRP;
(6) Change component part quantities in CAD;	(11) Modifying work centers status in MRP;

As an example, consider the partner policy for transaction "deleting a part in the CAD database". We can specify and decompose the partner policy, and model the transactions using Petri Nets as the following.

(1) The partner policy description

CAD partners have the sole authority to update the CAD databases. This is because the CAD user is the CAD database owner and thus is responsible for CAD data manipulation according to the data ownership protocol. The CAD user is also responsible for notification of any data updates in the CAD database.

(2) Policy decomposition

The generic policy can be decomposed into sub-descriptions:

(i) To delete a part in the CAD database, the CAD user needs to provide the following: part ID and related part IDs (e.g., the IDs of the parent part and child part).

The system needs to check that the provided part and relevant parts do exist in the CAD database system. If the answer is yes, the system sends votes and a brief message to the CAPP site and the MRP site, if the answer is no, the system outputs an error message and restarts;

(ii) at the CAPP site, the system needs to check the CAPP database for the existence of the specified part ID, and replies a positive vote if it does exist.

(iii) at the MRP site, the system performs the same actions.

(iv) if all positive votes are received, the local CAD site deletes part #3211, and sends "commit" command to servers at the CAPP and MRP sites, else it sends "abort" command. The CAPP and MRP partners will need to do appropriate adjustments in their responsible domains.

(3) Modeling the primitive transaction using Petri Nets (PN)

The PN model is shown in Figure 7, we use SysA, SysB and SysC to represent systems in sites A, B and C.

Figure 7 around here

6.2 Petri Nets Refinement and Verification

From the above example, we can see Petri Nets can model partner policies at various levels. All these PNs need be verified by verification techniques to keep the consistency of the system and eliminate any existing redundancies. Current PN verification techniques include the general verification, structure verification and domain knowledge verification (Lin, 1991). General verification assures that each PN has the basic valid characteristics and includes techniques such as "Reachability Tree" and "PN invariants". Reachability Tree can be used to detect and eliminate deadlocks and undesired final states. The net invariants specify the constant attributes of Petri Nets, e.g., the total number of tokens between two different markings is a constant. So by checking whether the net invariant has been changed, we can check any invalid transitions.

Structural verification checks completeness and consistency of the hierarchy and includes completeness checking and consistency checking. Completeness verification checks existence of unfirable transitions and unreachable places. Unfirable transitions are transitions which will never fire and can be checked out with finite set of initial markings and the reachability tree. Unreachable places are the places which will never be marked. Consistency verification checks the existence of redundant transitions and subsumed transitions. Redundant transitions are transitions which have the same sets of input places and output places. Subsumed transitions are transitions which have the same output set but one has more input places (see also Murata, 1989).

Domain knowledge verification ensures that the model of the domain in the rule base complies with the real world semantics. It checks the semantic validity for the knowledge base by checking the existence of conflicting transitions and free choice transitions. Conflicting transitions are the transitions that have the same set of input places but have conflicting output places, and can be detected by checking the incidence matrices. Free choice transitions are the transitions that have the same input places but have different non-contradictory output places and can be detected by checking their incidence matrices, and resolved by considering the semantic meanings of these transitions.

Using the Petri Nets modeling, we can perform PN refinement and verification techniques on our sample model for the transaction "Deleting part in the CAD database". The refined model can be referred to in Song (1996).

6.3 Building the Expert Rules and Validation

After PN modeling and verification, we come up with a consistent model of partner transaction policies. Now, we need to map the PN model to expert rules stored in the knowledge base. For the time being, we use plain text files to represent the expert rules. The expert rule for the sample transaction can be referred to in Song (1996). The expert rules for all the transactions in Section 6.1 compose the knowledge base.

The next step is to debug and validate the expert rules developed earlier. We need to test these expert rules using the partner policy specifications for the manufacturing transactions listed in Section 6.1. If the expert rules are not validated, go back to step 1 in Section 6.1, and iterate the process until we get a consistent and validated knowledge base.

To summarize, a systematic approach for forming the knowledge base is presented in this section. Starting from generic partner policy specifications, we decompose them and use Petri Nets to model and verify partner policies and inter-partner protocols to form a consistent knowledge base used as guidelines for partner workflows. The

knowledge base, the information hierarchy and the transaction workflow hierarchy, form our agile manufacturing information system architecture, which is expected to support distributed partner interactions in the design and manufacturing domain. In next section, we will discuss the trial implementation of a prototype system using TCP/IP socket programming and World Wide Web applications.

7 Prototype System Implementation

The previous sections described the Virtual Information System for Agile Manufacturing (VISAM) framework and its detailed component modules. Based on this agile manufacturing information system model, we have implemented a prototype system for illustrative purposes.

The system has the following four modules: Partner Server, Partner Client, Partner Workflow Manager and the local Knowledge Base. These and some other assisting modules are as follows:

1. Web Graphic User Interface is developed using the HyperText Markup Language (HTML). Mosaic Forms are used to accept user queries.

   The Web GUI is implemented using HTML. It includes the information for each local partner in the homepage and partner query submitting forms in the next layer using Mosaic Forms.

2. Web Server is down loaded from the World Wide Web. Currently it is NCSA 1.5.1.

3. Workflow Manager is developed using cgi/perl script. It interprets the user query and interacts with the local knowledge base to form the partner workflow query, so as to keep the global data consistency. It then invokes the partner client to connect to the other partners.

   The Partner workflow manager accepts the user query written in Partner Query Language, converts it into workflows in the format of the Partner Workflow Language, and invokes the client to execute the workflow. Considering the actual environment, the workflow manager includes two parts:

   (1) The Web Server is developed using cgi/perl script. This part accepts query through the Web Browser, consults the knowledge base and converts it into workflows, and sends a signal to the Client Daemon.

   (2) the Client Daemon which listens to the network all the time. When it receives the signal, it invokes the client to execute the workflow, and it sends a signal back to the Web server after workflow is executed. Control is then switched back to the Web Server to send query results back to the user.

4. Partner Server is listening to the network all the time. It accepts the query, interacts with local DBMS to process the query and sends results back to the inquirer. The server is implemented using the Unix daemon. It is designed using socket programming. The idea is to create a socket, bind it with a high port number and let it listen to the network all the time. When a connection requirement is heard, it forks a child process to deal with the requirement, and it continues to listen to the network itself.

5. Partner Client is responsible for sending the subqueries to relevant partners, coordinating query execution and accepting the results. The workflow execution is implemented using the two-phase-commit protocol. The client is implemented using socket programming. As the temporary system coordinator, it receives

the partner workflow from the workflow manager, parses the query and send the subqueries and a vote to relevant parties. When all the positive votes are received, it sends "commit" command; otherwise it sends "abort" command.

8 Conclusion

In the emerging agile manufacturing industry, a manufacturer needs to have a flexible and dynamic environment to cooperate with other manufacturers and form virtual enterprises to produce customized products. Based on the information requirements in an agile enterprise, a virtual agile manufacturing information system framework has been developed. Under this framework, a Workflow Manager, together with the partner Knowledge Base which captures partner specific and enterprise wide information management policies, the local partner DBMS, and partner Server and Client, construct a flexible integrated system for each partner. This Partner UNit Information System (PUNIS) supports partner local information transactions and provides a dynamic channel for local manufacturer to plug in the agile manufacturing network. The interactions of member Partner Unit Information Systems form the virtual information system which controls the information flow within the virtual enterprise.

Based on some important AMIS requirements identified in the paper, we have developed the AMIS framework, designed the AMIS model and implemented a prototype system for demonstration purpose. Our main contributions include the following: (1) Development of an AMIS framework which provides interoperability between partner databases and assures data consistency among partner databases. (2) Design of the AMIS model which includes the information hierarchy, the transaction hierarchy and the knowledge base: (a) the information hierarchy represents virtual enterprise information using Object Oriented Methodology; (b) the transaction hierarchy uses a Partner Query Language to compile queries and a Partner Workflow Language to specify partner workflows; (c) the knowledge base is built based on the partner policies and protocols. The modeling and validation of transaction procedure knowledge is accomplished using Petri Nets. (3) Implementation of the prototype system, which integrates the distributed partner information using a Client-Server architecture. Through the process of framework development and prototype system implementation, we have come up with a thorough approach for building partner self-support systems, to plug into the agile manufacturing network and form virtual enterprises.

Even while the approach is sound, significant work must be accomplished in system development. Firstly, workflow and information management policies must be acquired by careful system study and by interviewing domain experts. Information modeling work would include identification of system objects and specification of the algorithms that develop composite objects. Secondly, network and data communication technologies are changing rapidly, and security issues are becoming increasingly important. These may require modifications to system level modules. With these in mind, further investigation needs to be conducted in making this approach viable for industrial use. Future research directions may include: (1) System performance evaluation and improvement to ensure the information system satisfies the real-time agile manufacturing activities. (2) System scope extension. To accommodate commercial engineering, management and accounting applications easily, they must be based on an open architecture paradigm. It is noted that most current CAD/MRP packages do not have explicit DataBase Management Systems. (3) System level improvements. Recent technologies such as CORBA and Java are revolutionizing distributed object communication, and powerful GUI development environments are available.

References

[1] Adiga, S. and Gadre, M., "Object-Oriented Software Modeling of a Flexible Manufacturing System," *Journal of Intelligent and Robotic Systems*, 3, 147-165 (1990).

[2] Berio, G., Di Leva, A., Giolito, P. and Vernadat, F., "The M*-Object Methodology for Information System Design in CIM Environments," *IEEE Transactions on Systems, Man, and Cybernetics*, 25, 1, 68-85 (1995).

[3] Boucher, T., Jafari, M. and Meredith, G., "Petri Net Control of an Automated Manufacturing Cell," *Advanced Manufacturing Engineering*, 2, 6, 151-157 (1990).

[4] Changchien, S., Lin, L. and Sun, D., "A Dynamic Control Model of Flexible Manufacturing Cells Using the Information Processing Object Hierarchy," accepted in *International Journal of Flexible Automation and Integrated Manufacturing*, 1994.

[5] Dilts, D.M. and Wu, W., "Using Knowledge-Based Technology to Integrate CIM Databases," *IEEE Transactions on Knowledge and Data Engineering*, 3, 2, 237-245 (1991).

[6] Elmasri, R. and Navathe, S.B., *Fundamentals of Database Systems*, Benjamin-Cummings, Redwood City, CA (1994)

[7] Flatau, U., "Design an Information System for Integrated Manufacturing Systems," *Design and Analysis of Integrated Manufacturing Systems*, Editor: W.D. Compton, National Academy Press, Washington DC, 60-78 (1988).

[8] Harhalakis, G., Lin, C.-P., Mark, L. and Muro-Medrano, P., "Structured Representation of Rule-Based Specifications in CIM Using Updated Petri-Nets," *IEEE transactions on Systems, Man and Cybernetics*, 25, 1, 130-144 (1995).

[9] Howard, H. and Rehak, H., "Kadbase: A Prototype Expert System-database Interface for Engineering Systems'," *IEEE Expert*, 4, 3, 169-181 (1985).

[10] Hsu, C. and Skevington, C., "Integration of Data and Knowledge in Manufacturing enterprises: a conceptual framework," *Journal of Manufacturing Systems*, 6, 4, 277-285 (1987).

[11] Hsu, C. and Rattner L., "Information Modeling for Computerized Manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics*, 20, 4, 758-776 (1990).

[12] Krishnamurthy, V., Su, Y.W., Lam, H., Mitchell, M. and Barkmeyer, E., "IMDAS–An Integrated Manufacturing Data Administration System," *Data Knowledge Engineering*, 3, 4, 109-131 (1988).

[13] Law, K., Wiederhold, G., Siambela, N., Sujansky, W., Zingmond, D., Singh, H. and Barsaloou, T., "Architecture for Managing Design Objects in a Shareable Relational Framework," *Int. J. of Systems Automation: Research and Applications*, 1, 47-65 (1991).

[14] Lin, C.-P., "Design, Verification and Implementation of Rule-Based System for Integrated Manufacturing," Ph.D. dissertation, University of Maryland, (1991).

[15] Mowbray, J. and Zahavi, R., *The ESSENTIAL CORBA: System Integration Using Distributed Objects*, John Wiley and Object Management Group (1995).

[16] Murata, T., "Petri Nets: Properties, analysis and applications," *Proceedings of the IEEE*, 77, 4, 541-580 (1989).

[17] Nagel, R. and Dove, R., "21st Century Manufacturing Enterprise Strategy," Iacocca Institute, Lehigh University, Bethlehem, PA (1993).

[18] Pandiarajan, V. and Patun, R., "Agile Manufacturing Initiatives at Concurrent Technologies Corp.," *Industrial Engineering*, 26, 2, 46-49 (1994).

[19] Peterson, J., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ (1981).

[20] Qiao, L., Zhang, C., Liu, T., Wang, H. and Fisher, G., "A PDES/STEP Based Product Data Preparation Procedure for Computer-Aided Process Planning," *Computer in Industry*, 21, 11-22 (1993).

[21] Rusinkiewicz, M., Sheth, A. and Karabatis, G., "Specifying Interdatabase Dependencies in a Multidatabase Environment," *Computer*, 24, 12, 46-52 (1995).

[22] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ (1991).

[23] Song, L., "Design and Implementation of an Agile Manufacturing Information System for Agile Manufacturing Enterprises," Ph.D. dissertation, Department of Industrial Engineering, SUNY-Buffalo, (1996).

[24] Spooner, D.L. and Hardwick, M., "Using Views in an Information Infrastructure for Manufacturing," *Proc. of the Agile and Intelligent Manufacturing Symposium*, RPI, Troy, NY, CD-Rom (1996).

[25] Sriram, D., Ahmed, S. and Logcher, R., "A Transaction Management Framework for Collaborative Engineering," *Engineering with Computers*, 8, 213-232 (1992).

[26] Sriram, D., Logcher, R., Wong, A. and Ahmed, S., "Computer Aided Cooperative Product Development: A Case Study," *Int. J. of Systems Automation: Research and Applications*, 1, 89-112 (1991).

[27] Sun, D. and Lin, L., "A Framework for Object-Oriented Simulation Model Building," *Proceedings of the Twenty-Third Pittsburgh Conference of Modeling and Simulation*, University of Pittsburgh, 23, 2, 1055-1062 (1992).

[28] Teng, S. and Black, J., "Cellular Manufacturing Systems Modeling: The Petri Net Approach," *Journal of Manufacturing Systems*, 9, 1, 45-54 (1990).

[29] Tiwari, S. and Howard, H., "Distributed AEC Databases for Collaborative Design," *Engineering with Computers*, 10, 140-154 (1994).

[30] Zhang, A., "Impact of Multimedia Data on Workflows," *CSCW-94 Workshop on Distributed Systems, Multimedia, and Infrastructure Support in CSCW*, also in *ACM SIGOIS Bulletin*, 15, 2, 57-58 (1994).

List of Figures

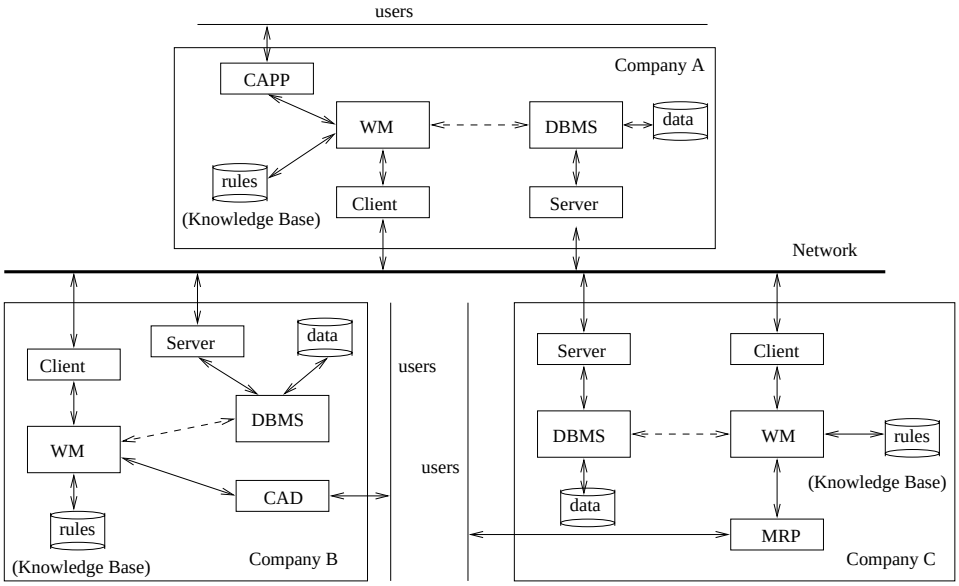


Figure 1: A Virtual Enterprise and Information System Model

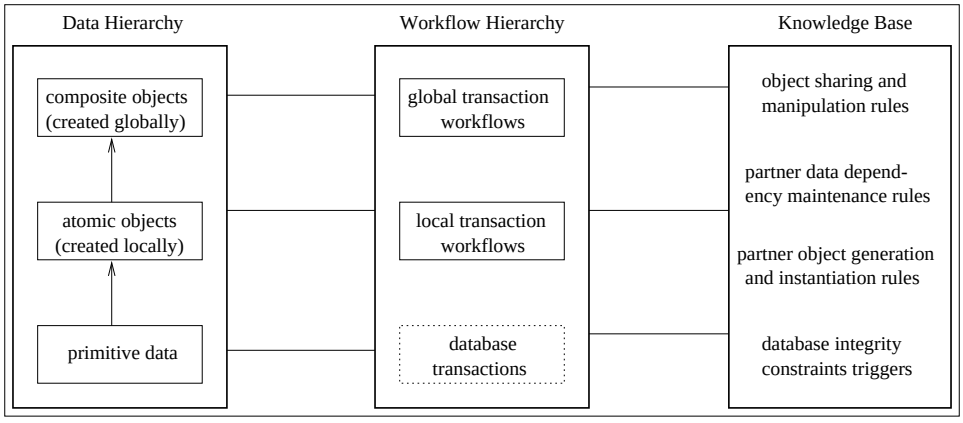


Figure 2: Two Hierarchies and the Knowledge Base

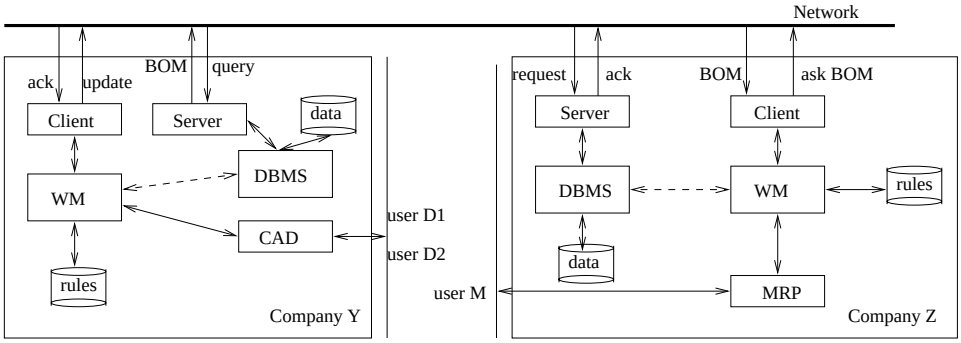


Figure 3: An Operational Scenario

```
consturct POR (using POR_FORMATION)

from BOM in CAD, GR in SALES and
 INVENTORY in MRP

where partID=32;
```

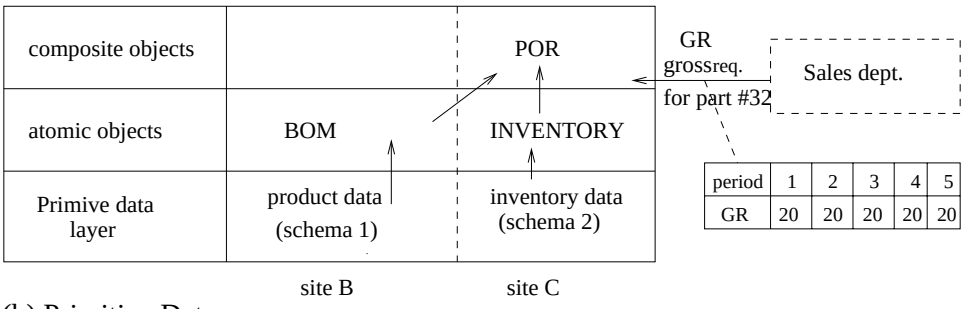
(PQL)

```
beginprogram
  object POR of
  TimePeriod:      integer
  PartID:          integer
  OrderQuantity:   integer
  endobject
     subtransactions:
       subtransaction t1: partner B
          construct BOM from CAD
       subtransaction t2: partner C
          construct INVENTORY from MRP
       subtransaction t3:
          Read GR from Sales
       subtransaction t4:
          Construct POR from GR, BOM and IN-
          VENTORY using POR_FORMATION
       dependency:
          t4 depends on t1, t2, and t3
          POR: output
endprogram
```

(PWL)

Figure 4: A sample partner query compiled using PQL and PWL

(a) Data Hierarchy



(b) Primitive Data

schema 1

table 1: Part #32_structure

part_id	321	3211
quantity	2	1

table 2: Part #321_structure

part_id	3211
quantity	1

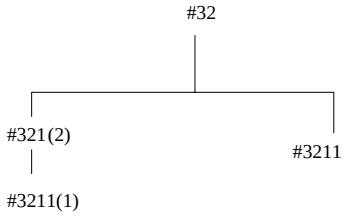
schema 2

table 3: inventory_data

part_id	inventory
32	41
321	52
3211	60

(c) Atomic Object: (BOM for Part #32)

Notation: 32-1(321-2(3211-1), 3211-1)



(d) Composite Object (POR for Part #32)

timeperiod	1	2	3	4	5
#32		25	25	25	
#321	50	50			
#3211	65	25	25		

Figure 5: A Simplified Data Hierarchy

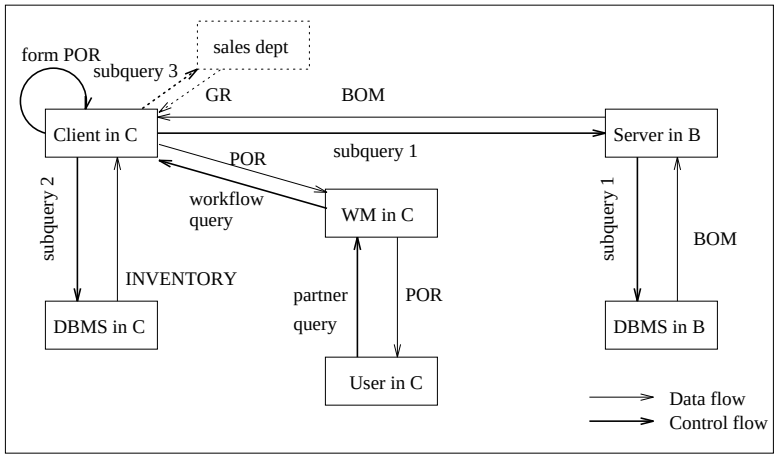
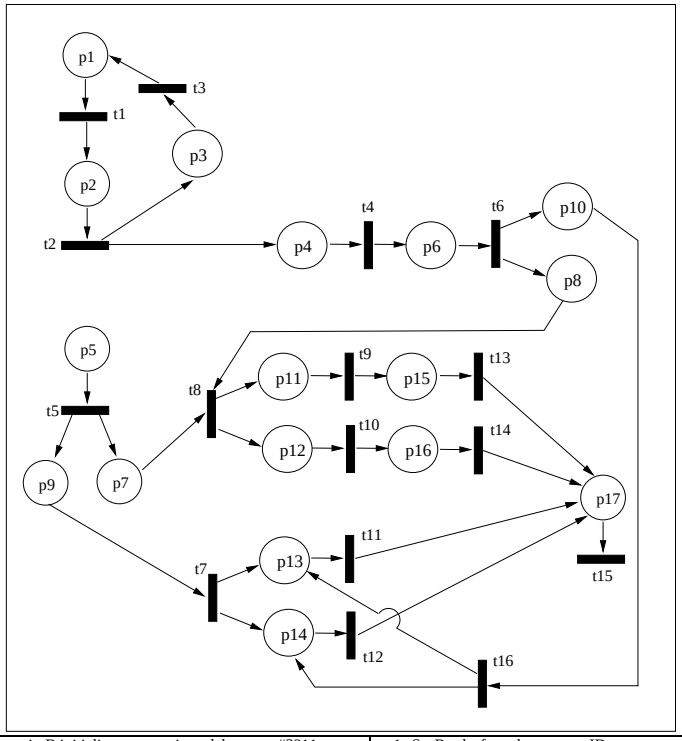


Figure 6: Data flow and Control Flow Diagram for the sample workflow



p1: CAD user in B initializes transaction: delete part #3211;	t1: SysB asks for relevant part IDs;
p2: required information is provided;	t2: SysB check existence of ID#;
p3: ID does not exist;	t3: SysB outputs error message, restarts;
p4: ID exists in CAD database;	t4: SysB sends votes to A and C;
p5: vote is received by A;	t5: SysA checks existence of part ID;
p6: vote is received by C;	t6: SysC checks existence of part ID;
p7: ID does exist in CAPP at A, and A is ready;	p8: ID does exist in MRP at C, and C is ready;
p9: ID does not exist in CAPP at A;	p10: ID does not exist in MRP at C;
t7, t16: client sends “abort” and ends process with a reply to user;	t8: SysB delete the part, and sends “commit” to A and C;
p11: server in A receives “commit”;	t9: sever in A sends subquery to DBMS in A;
p12: server in C receives “commit”;	t10: sever in C sends subquery to DBMS in C;
p13: server in A receives “abort”;	t11: sever in A aborts query;
p14: server in C receives “abort”;	t12: sever in C aborts query;
p15: DBMS in A receives command;	t13: DBMS in A puts part on “hold” and replies;
p16: DBMS in C receives command;	t14: DBMS in C puts part on “hold” and replies;
p17: result is received by SysB;	t15: Sys B ends proc. and replies to user.

Figure 7: Petri Net Model for Transaction “Delete Part in CAD”